

Classification on MNIST Dataset

In the question, you are asked to write code for classifying image data using various classifiers. The [MNIST dataset](#) is a database of 70,000 28×28 pixel grayscale images of handwritten single digits.

A Few Task Reminders:

- Import required libraries.
- Load the data and split into train, validation, and test sets. You can also perform k-fold cross-validation on the train set for better performance estimates and nested cross-validation for hyperparameter tuning.
- Perform any required data pre-processing.
- Train K-NN, Logistic Regression, Decision Trees, and SVM on the data.
- Make predictions, evaluate, and compare the models. Generate confusion matrices and classification reports.
- Summarize your findings and make sure you sufficiently document your code.

Explore the different classifiers listed above and perform hyperparameter tuning as follows:

- For K-NN, explore the effect of varying the number of nearest neighbors
- For Logistic Regression, explore the effect of varying regularization parameter
- For Decision Trees, explore the effect of varying the max depth of the tree
- For SVM, explore the effect of varying the penalty parameter and kernel function

Note. It is intentional that this problem assignment extends outside of what we have covered in class (i.e. text data pre-processing) and encourages more independent learning and exploration with ML hands-on experience and applications. I hope you would have fun with these type of questions and that they are not very stressful. Also, feedback is welcomed and encouraged!

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
```

```
In [12]: # Fetch MNIST data (might take some time)
mnist = fetch_openml('mnist_784')

X = mnist.data.astype('float32')
y = mnist.target.astype('int64')

# Normalize the data
X /= 255.0
```

```
In [144... from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.model_selection import GridSearchCV, train_test_split
```

```

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report, confusion_matrix
import pickle

```

```

In [76]: scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)

```

```

In [77]: X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.2)

```

```

In [78]: pca=PCA(0.95)
pca.fit(X_train)

```

```

Out[78]: PCA(n_components=0.95)

```

```

In [79]: X_train_pca=pca.transform(X_train)
X_test_pca=pca.transform(X_test)

```

```

In [164... search_space_k={
    'n_neighbors' : [5,10,15,20]
}
search_space_l={
    'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
}
search_space_t={
    'max_depth': [None, 5,10,15,20],
}
search_space_s={
    'kernel' : ['linear', 'poly', 'rbf', 'sigmoid']
}
search_space_lin_svc={
    'C' : [0.001, 0.01, 0.1, 1]
}

```

```

In [81]: knn=KNeighborsClassifier()
clf_knn=GridSearchCV(knn,search_space_k,cv=3,n_jobs=-1)
clf_knn.fit(X_train_pca,y_train)

```

```

Out[81]: GridSearchCV(cv=3, estimator=KNeighborsClassifier(), n_jobs=-1,
    param_grid={'n_neighbors': [5, 10, 15, 20]})

```

```

In [98]: clf_knn.best_params_

```

```

Out[98]: {'n_neighbors': 5}

```

```

In [84]: clf_knn.cv_results_

```

```

Out[84]: {'mean_fit_time': array([0.92393788, 0.92772937, 0.91927274, 1.54400261]),
    'std_fit_time': array([0.46382901, 0.46329492, 0.45987636, 0.20259877]),
    'mean_score_time': array([161.5054493, 170.73705546, 161.42259161, 90.773199

```

```

]),
'std_score_time': array([18.22641628,  4.96037859, 18.59974226, 53.06128924]),
'param_n_neighbors': masked_array(data=[5, 10, 15, 20],
                                   mask=[False, False, False, False],
                                   fill_value='?',
                                   dtype=object),
'params': [{'n_neighbors': 5},
            {'n_neighbors': 10},
            {'n_neighbors': 15},
            {'n_neighbors': 20}],
'split0_test_score': array([0.94166176, 0.93817968, 0.93437617, 0.9305191 ]),
'split1_test_score': array([0.94476884, 0.94048321, 0.93651899, 0.93314405]),
'split2_test_score': array([0.93887282, 0.93405122, 0.92997964, 0.92714026]),
'mean_test_score': array([0.94176781, 0.93757137, 0.93362493, 0.9302678 ]),
'std_test_score': array([0.00240821, 0.00266085, 0.00272201, 0.00245747]),
'rank_test_score': array([1, 2, 3, 4], dtype=int32)}

```

```
In [87]: predicted_knn=clf_knn.predict(X_test_pca)
```

```
In [121... print(classification_report(y_test,predicted_knn))
```

	precision	recall	f1-score	support
0	0.97	0.98	0.98	1362
1	0.97	0.99	0.98	1624
2	0.96	0.95	0.95	1390
3	0.93	0.96	0.95	1415
4	0.96	0.95	0.95	1361
5	0.93	0.92	0.93	1271
6	0.96	0.97	0.96	1391
7	0.94	0.96	0.95	1413
8	0.97	0.91	0.94	1372
9	0.93	0.94	0.94	1401
accuracy			0.95	14000
macro avg	0.95	0.95	0.95	14000
weighted avg	0.95	0.95	0.95	14000

```
In [125... confusion_matrix(y_test,predicted_knn)
```

```

Out[125... array([[1339,  0,  2,  2,  1,  3, 14,  0,  0,  1],
 [  0, 1607,  2,  1,  0,  2,  6,  4,  1,  1],
 [ 13,  6, 1316, 18,  4,  0,  8, 17,  5,  3],
 [  3,  4,  8, 1358,  1, 15,  1, 12,  5,  8],
 [  0,  9, 13,  2, 1289,  3,  5,  9,  2, 29],
 [  5,  2,  5, 39,  5, 1174, 20,  2,  9, 10],
 [  8,  5,  6,  1,  7,  10, 1349,  0,  5,  0],
 [  0,  8,  8,  4, 10,  0,  0, 1353,  3, 27],
 [  8, 18, 11, 21,  5, 44,  2,  7, 1243, 13],
 [  4,  3,  6, 13, 22,  6,  0, 33,  2, 1312]])

```

```

In [107... with open('model_knn.pkl','wb') as f:
            pickle.dump(clf_knn,f)

#with open('model_knn.pkl','rb') as f:
#    clf2=pickle.load(f)

```

```
In [173... logr=LogisticRegression(solver='sag',max_iter=1000,tol=1e-3)
clf_logr=GridSearchCV(logr,search_space_l,cv=3,n_jobs=-1)
clf_logr.fit(X_train_pca,y_train)
```

```
Out[173... GridSearchCV(cv=3,
              estimator=LogisticRegression(max_iter=1000, solver='sag',
                                             tol=0.001),
              n_jobs=-1, param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]})
```

```
In [174... clf_logr.best_params_
```

```
Out[174... {'C': 1}
```

```
In [175... clf_logr.cv_results_
```

```
Out[175... {'mean_fit_time': array([ 25.23615702,  80.89298693, 120.65007218, 127.03134791,
                             121.07482306, 113.22952151, 106.64190173]),
          'std_fit_time': array([0.76727742, 2.92715113, 7.1853959 , 7.42636872, 3.819667
43,
                                4.16910999, 5.81419394]),
          'mean_score_time': array([0.03298338, 0.04689233, 0.03929575, 0.03024785, 0.016
72188,
                                0.01860929, 0.0159061 ]),
          'std_score_time': array([0.01025135, 0.01655264, 0.00589005, 0.01312707, 0.0012
6374,
                                0.00218765, 0.00259891]),
          'param_C': masked_array(data=[0.001, 0.01, 0.1, 1, 10, 100, 1000],
                                  mask=[False, False, False, False, False, False, False],
                                  fill_value='?',
                                  dtype=object),
          'params': [{'C': 0.001},
                     {'C': 0.01},
                     {'C': 0.1},
                     {'C': 1},
                     {'C': 10},
                     {'C': 100},
                     {'C': 1000}],
          'split0_test_score': array([0.90989447, 0.91825146, 0.92098355, 0.92087641, 0.9
2098355,
                                     0.92092998, 0.92092998]),
          'split1_test_score': array([0.91123373, 0.91964429, 0.92039428, 0.92066213, 0.9
2060856,
                                     0.92055499, 0.92055499]),
          'split2_test_score': array([0.90849673, 0.91808636, 0.91942569, 0.91969356, 0.9
1958641,
                                     0.91963999, 0.91963999]),
          'mean_test_score': array([0.90987498, 0.9186607 , 0.92026784, 0.9204107 , 0.920
39284,
                                     0.92037499, 0.92037499]),
          'std_test_score': array([0.00111746, 0.00069876, 0.00064225, 0.00051459, 0.0005
9042,
                                   0.0005418 , 0.0005418 ]),
          'rank_test_score': array([7, 6, 5, 1, 2, 3, 3], dtype=int32)}
```

```
In [176... predicted_logr=clf_logr.predict(X_test_pca)
```

In [177...

```
print(classification_report(y_test,predicted_logr))
```

	precision	recall	f1-score	support
0	0.96	0.97	0.96	1362
1	0.95	0.97	0.96	1624
2	0.93	0.90	0.91	1390
3	0.91	0.89	0.90	1415
4	0.92	0.95	0.94	1361
5	0.90	0.87	0.88	1271
6	0.94	0.95	0.94	1391
7	0.93	0.94	0.93	1413
8	0.89	0.89	0.89	1372
9	0.90	0.91	0.90	1401
accuracy			0.92	14000
macro avg	0.92	0.92	0.92	14000
weighted avg	0.92	0.92	0.92	14000

In [178...

```
confusion_matrix(y_test,predicted_logr)
```

```
Out[178...] array([[1318, 1, 2, 4, 2, 8, 12, 6, 6, 3],
      [ 0, 1574, 6, 8, 2, 6, 3, 8, 15, 2],
      [ 9, 15, 1250, 16, 14, 8, 16, 19, 36, 7],
      [ 7, 7, 38, 1258, 3, 39, 8, 10, 28, 17],
      [ 0, 5, 6, 2, 1290, 3, 12, 3, 5, 35],
      [18, 6, 6, 43, 14, 1102, 30, 3, 40, 9],
      [15, 4, 10, 1, 14, 18, 1318, 5, 5, 1],
      [ 0, 8, 10, 4, 11, 2, 0, 1330, 3, 45],
      [10, 36, 10, 28, 7, 34, 6, 5, 1215, 21],
      [ 3, 7, 5, 20, 39, 7, 0, 43, 7, 1270]])
```

In [179...

```
with open('model_logr.pkl','wb') as f:
    pickle.dump(clf_logr,f)
```

In [110...

```
decision_tree=DecisionTreeClassifier()
clf_dct=GridSearchCV(decision_tree,search_space_t,n_jobs=-1)
clf_dct.fit(X_train_pca,y_train)
```

Out[110...

```
GridSearchCV(estimator=DecisionTreeClassifier(), n_jobs=-1,
              param_grid={'max_depth': [None, 5, 10, 15, 20]})
```

In [111...

```
clf_dct.best_params_
```

Out[111...

```
{'max_depth': 15}
```

In [112...

```
clf_dct.cv_results_
```

Out[112...

```
{'mean_fit_time': array([37.50029001, 10.53001604, 18.43235664, 24.73020592, 26.
58777823]),
 'std_fit_time': array([1.01606573, 0.06931991, 0.10673776, 0.07807035, 0.250087
74]),
 'mean_score_time': array([0.01499977, 0.01235256, 0.00611401, 0.00481806, 0.005
67966]),
 'std_score_time': array([0.0073633 , 0.0039344 , 0.00179303, 0.0007732 , 0.0005
```

```

2917])),
'param_max_depth': masked_array(data=[None, 5, 10, 15, 20],
                                mask=[False, False, False, False, False],
                                fill_value='?',
                                dtype=object),
'params': [{'max_depth': None},
            {'max_depth': 5},
            {'max_depth': 10},
            {'max_depth': 15},
            {'max_depth': 20}],
'split0_test_score': array([0.81169643, 0.58803571, 0.79258929, 0.820625 , 0.8
1482143]),
'split1_test_score': array([0.81214286, 0.59535714, 0.80125 , 0.82294643, 0.8
1410714]),
'split2_test_score': array([0.819375 , 0.59642857, 0.80723214, 0.82705357, 0.8
2133929]),
'split3_test_score': array([0.81803571, 0.59678571, 0.80392857, 0.82482143, 0.8
2276786]),
'split4_test_score': array([0.81410714, 0.60080357, 0.80178571, 0.82535714, 0.8
19375 ]),
'mean_test_score': array([0.81507143, 0.59548214, 0.80135714, 0.82416071, 0.818
48214]),
'std_test_score': array([0.00310499, 0.00415638, 0.00486206, 0.00220056, 0.0034
6033]),
'rank_test_score': array([3, 5, 4, 1, 2], dtype=int32)}

```

```
In [115... predicted_dct=clf_dct.predict(X_test_pca)
```

```
In [122... print(classification_report(y_test,predicted_dct))
```

	precision	recall	f1-score	support
0	0.89	0.89	0.89	1362
1	0.95	0.95	0.95	1624
2	0.83	0.82	0.83	1390
3	0.80	0.77	0.78	1415
4	0.79	0.82	0.81	1361
5	0.76	0.74	0.75	1271
6	0.89	0.89	0.89	1391
7	0.83	0.86	0.84	1413
8	0.74	0.75	0.75	1372
9	0.79	0.78	0.78	1401
accuracy			0.83	14000
macro avg	0.83	0.83	0.83	14000
weighted avg	0.83	0.83	0.83	14000

```
In [128... confusion_matrix(y_test,predicted_dct)
```

```

Out[128... array([[1208,  2,  24,  15,  10,  35,  24,  12,  23,  9],
 [  1, 1542,  10,  8,  12,  7,  8,  13,  19,  4],
 [ 24,  14, 1140,  53,  27,  16,  35,  26,  43, 12],
 [ 23,  20,  53, 1086,  11,  81,  6,  29,  88, 18],
 [ 13,  7,  19,  4, 1115,  29,  18,  44,  27, 85],
 [ 30,  13,  15,  69,  24,  944,  40,  16,  95, 25],
 [ 23,  4,  32,  4,  29,  22, 1244,  3,  20, 10],
 [  8,  6,  19,  17,  34,  9,  4, 1209,  19, 88],
 [ 17,  19,  40,  74,  32,  77,  21,  16, 1035, 41],
 [ 14,  4,  15,  26, 112,  21,  1,  91,  29, 1088]])

```

```
In [117... with open('model_dct.pkl','wb') as f:
            pickle.dump(clf_dct,f)
```

```
In [157... lsvc=LinearSVC(dual=False)
clf_lsvc=GridSearchCV(lsvc,search_space_lin_svc,n_jobs=-1,cv=2)
clf_lsvc.fit(X_train,y_train)
```

```
Out[157... GridSearchCV(cv=2, estimator=LinearSVC(dual=False), n_jobs=-1,
              param_grid={'C': [0.001, 0.01, 0.1, 1]})
```

```
In [158... clf_lsvc.best_params_
```

```
Out[158... {'C': 0.01}
```

```
In [159... clf_lsvc.cv_results_
```

```
Out[159... {'mean_fit_time': array([ 161.88624752, 1146.01064491, 1940.61772645, 3071.01824
355]),
'std_fit_time': array([ 5.47511542, 15.340451 , 13.37943447, 27.57126641]),
'mean_score_time': array([0.07130957, 0.06468999, 0.10819721, 0.10718298]),
'std_score_time': array([0.00231433, 0.00124609, 0.00077796, 0.00109005]),
'param_C': masked_array(data=[0.001, 0.01, 0.1, 1],
                        mask=[False, False, False, False],
                        fill_value='?',
                        dtype=object),
'params': [{'C': 0.001}, {'C': 0.01}, {'C': 0.1}, {'C': 1}],
'split0_test_score': array([0.89728571, 0.90539286, 0.90496429, 0.90235714]),
'split1_test_score': array([0.89375 , 0.90242857, 0.90132143, 0.89907143]),
'mean_test_score': array([0.89551786, 0.90391071, 0.90314286, 0.90071429]),
'std_test_score': array([0.00176786, 0.00148214, 0.00182143, 0.00164286]),
'rank_test_score': array([4, 1, 2, 3], dtype=int32)}
```

```
In [160... predicted_lsvc=clf_lsvc.predict(X_test)
```

```
In [161... print(classification_report(y_test,predicted_lsvc))
```

	precision	recall	f1-score	support
0	0.94	0.97	0.96	1362
1	0.94	0.97	0.95	1624
2	0.92	0.88	0.90	1390
3	0.89	0.88	0.89	1415
4	0.91	0.94	0.92	1361
5	0.90	0.85	0.87	1271
6	0.93	0.95	0.94	1391
7	0.92	0.95	0.94	1413
8	0.88	0.86	0.87	1372
9	0.90	0.88	0.89	1401
accuracy			0.91	14000
macro avg	0.91	0.91	0.91	14000
weighted avg	0.91	0.91	0.91	14000

```
In [162... confusion_matrix(y_test,predicted_lsvc)
```

```
Out[162...] array([[1327, 1, 3, 2, 3, 5, 11, 3, 4, 3],
      [ 1, 1577, 6, 7, 2, 5, 5, 6, 14, 1],
      [ 12, 21, 1229, 22, 15, 9, 14, 20, 41, 7],
      [ 10, 13, 40, 1245, 3, 31, 8, 9, 37, 19],
      [ 3, 5, 7, 3, 1278, 4, 13, 4, 5, 39],
      [ 18, 4, 10, 53, 21, 1075, 34, 4, 39, 13],
      [ 14, 5, 13, 0, 10, 15, 1321, 4, 8, 1],
      [ 0, 6, 9, 4, 9, 2, 4, 1339, 4, 36],
      [ 13, 42, 11, 33, 12, 43, 12, 8, 1176, 22],
      [ 7, 9, 6, 26, 51, 10, 0, 52, 9, 1231]])
```

```
In [163...] with open('model_lsvc.pkl', 'wb') as f:
              pickle.dump(clf_lsvc, f)
```

```
In [166...] svc=SVC(C=0.001, max_iter=2000)
              clf_svc=GridSearchCV(svc, search_space_s, n_jobs=-1, cv=5)
              clf_svc.fit(X_train_pca, y_train)
```

/opt/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:255: Convergence Warning: Solver terminated early (max_iter=2000). Consider pre-processing your data with StandardScaler or MinMaxScaler.

warnings.warn('Solver terminated early (max_iter=%i).')

```
Out[166...] GridSearchCV(cv=5, estimator=SVC(C=0.001, max_iter=2000), n_jobs=-1,
                        param_grid={'kernel': ['linear', 'poly', 'rbf', 'sigmoid']})
```

```
In [167...] clf_svc.best_params_
```

```
Out[167...] {'kernel': 'linear'}
```

```
In [168...] clf_svc.cv_results_
```

```
Out[168...] {'mean_fit_time': array([ 171.95512977,  984.58973093, 1038.45634322,  751.77450
013]),
              'std_fit_time': array([ 3.13967254,  6.40468162, 19.94938339,  8.52851972]),
              'mean_score_time': array([ 27.29575744,  59.96360068, 110.68338299,  49.6229634
3]),
              'std_score_time': array([0.26372807, 0.38421839, 4.67408844, 2.68643461]),
              'param_kernel': masked_array(data=['linear', 'poly', 'rbf', 'sigmoid'],
                                           mask=[False, False, False, False],
                                           fill_value='?',
                                           dtype=object),
              'params': [{'kernel': 'linear'},
                         {'kernel': 'poly'},
                         {'kernel': 'rbf'},
                         {'kernel': 'sigmoid'}],
              'split0_test_score': array([0.93633929, 0.09928571, 0.63946429, 0.59705357]),
              'split1_test_score': array([0.93892857, 0.09991071, 0.62598214, 0.59776786]),
              'split2_test_score': array([0.94151786, 0.09946429, 0.63098214, 0.59955357]),
              'split3_test_score': array([0.93464286, 0.09955357, 0.639375, 0.60741071]),
              'split4_test_score': array([0.93857143, 0.09946429, 0.63017857, 0.61330357]),
              'mean_test_score': array([0.938, 0.09953571, 0.63319643, 0.60301786]),
              'std_test_score': array([0.00234915, 0.00020671, 0.00535756, 0.00632818]),
              'rank_test_score': array([1, 4, 2, 3], dtype=int32)}
```

```
In [169...] predicted_svc=clf_svc.predict(X_test_pca)
```


In [170...

```
print(classification_report(y_test,predicted_svc))
```

	precision	recall	f1-score	support
0	0.96	0.99	0.97	1362
1	0.96	0.98	0.97	1624
2	0.94	0.93	0.93	1390
3	0.93	0.91	0.92	1415
4	0.94	0.96	0.95	1361
5	0.92	0.90	0.91	1271
6	0.96	0.96	0.96	1391
7	0.95	0.95	0.95	1413
8	0.93	0.92	0.92	1372
9	0.94	0.92	0.93	1401
accuracy			0.94	14000
macro avg	0.94	0.94	0.94	14000
weighted avg	0.94	0.94	0.94	14000

In [171...

```
confusion_matrix(y_test,predicted_svc)
```

```
Out[171...] array([[1342,  0,  2,  1,  3,  2,  7,  0,  5,  0],
       [  1, 1586,  8,  5,  3,  1,  4,  6,  8,  2],
       [ 15,  10, 1298, 12, 12,  3, 11, 11, 16,  2],
       [  3,  8,  31, 1285,  2, 38,  4,  9, 29,  6],
       [  3,  3, 10,  0, 1313,  1,  5,  2,  2, 22],
       [ 11,  7,  5, 39,  7, 1149, 20,  1, 26,  6],
       [ 13,  3,  7,  1,  7,  15, 1341,  1,  3,  0],
       [  0,  5, 10,  5, 10,  0,  0, 1346,  1, 36],
       [  8, 25,  9, 19,  8, 29,  6,  6, 1257,  5],
       [  3,  6,  7, 19, 35,  5,  0, 33,  8, 1285]])
```

In [172...

```
with open('model_svc.pkl','wb') as f:
    pickle.dump(clf_svc,f)
```

In []: